Heuristic Pattern Search for Bound Constrained Minimax Problems

Isabel A. C. P. Espírito Santo and Edite M. G. P. Fernandes

Algoritmi R & D Centre, University of Minho, 4710-057 Braga, Portugal
{iapinho;emgpf}@dps.uminho.pt
WWW home page: http://www.norg.uminho.pt

Abstract. This paper presents a pattern search algorithm and its hybridization with a random descent search for solving bound constrained minimax problems. The herein proposed heuristic pattern search method combines the Hooke and Jeeves (HJ) pattern and exploratory moves with a randomly generated approximate descent direction. Two versions of the heuristic algorithm have been applied to several benchmark minimax problems and compared with the original HJ pattern search algorithm.

Keywords: minimax problems, Hooke and Jeeves, heuristic pattern search, hybridization, random descent search

1 Minimax problems

In general, a bound constrained finite minimax problem can be defined as

$$\underset{x\in\Omega\subset\mathbb{R}^n}{\text{minimize}}\, f(x), \text{ where } f(x) = \max_{j=1,\ldots,m} F_j(x), \tag{1}$$

$F_j : \mathbb{R}^n \to \mathbb{R},\ j = 1,\ldots,m$ are continuously differentiable functions and $\Omega = \{x \in \mathbb{R}^n : l \le x \le u\}$. These problems have been difficult to solve through traditional gradient based algorithms, since the first derivatives of $f(x)$ are discontinuous at points where $f(x) = F_j(x)$ for two or more values of j in the set $\{1,\ldots,m\}$, even if all the functions $F_j(x)$ have continuous first derivatives. This type of problems appears in many engineering areas, such as, optimal control, engineering design, discrete optimization, Chebyshev approximation and game theory applications. For a more thorough review of applications the reader is referred to [7, 17] and to the references therein listed. To solve a problem like (1), a common strategy adapts a smoothing technique which consists of solving a sequence of smooth problems that approximate the minimax problem in the limit [10, 14, 17]. Choosing an updating rule for the smoothing parameter may be problematic. The algorithms based on these smooth techniques aim to generate a sequence of approximations that converges to a Kuhn-Tucker point of the minimax problem (1), for a decreasing sequence of positive smoothing parameters. However, these parameters may become rather small too fast and the smooth problems become significantly ill-conditioned. A different approach to obtain a solution to (1) considers solving an equivalent differentiable nonlinear programming problem

$$\underset{x\in\Omega\subset\mathbb{R}^n, z\in\mathbb{R}}{\text{minimize}}\, z, \text{ s.t. } F_j(x) - z \le 0,\ j = 1,\ldots,m. \tag{2}$$

Several techniques have been proposed to solve (2). In [2] a continuously differentiable exact penalty function is constructed for problem (2) and a gradient based method is applied to the penalty function. More recently, [18] and [19] use a similar approach. In the former, a trust-region Newton conjugate gradient algorithm is proposed. In the latter paper, an improved SQP algorithm is presented.

Other popular derivative-free numerical methods for solving problem (1) are stochastic-type algorithms. A swarm intelligence algorithm that has been extensively used in this context is the particle swarm optimization (PSO), see for example [7, 12, 13]. There is however a well-known problem regarding the accuracy of the solutions found by this type of algorithms. They can detect the region of attraction of the global minimizers fast but they are not capable of reaching the solution with high precision. Further, being population-based methods, they are computationally expensive. Recently, hybrid algorithms use stochastic methods, for global search, and popular gradient techniques as a local search method [15]. However, gradient-based strategies are not as appropriate as derivative-free methods for solving problems like (1). The hybridization of PSO with a random walk for local search is proposed in [13].

Derivative-free methods like the generalized pattern search approach [16] and the Hooke and Jeeves search [6] have been used for solving nonsmooth optimization problems. However, they may not be able to reach the solution in some particular problems. In this paper we aim to present a deterministic method with a local random search hybridization. The adopted approach for solving problem (1) relies on a popular derivative-free method, known as the Hooke and Jeeves pattern search method for bound constrained minimization [6, 8], and a simple heuristic that generates a random descent direction. Two different schemata are proposed and tested. Good accuracy solutions and reductions on the number of objective function evaluations are obtained when compared with the original Hooke and Jeeves search.

The remainder of the paper is organized as follows. Section 2 briefly introduces the original Hooke and Jeeves pattern search method for bound constrained optimization, Section 3 is devoted to describe our main ideas behind the pattern search hybridization with a descent search and Section 4 contains the numerical results. Section 5 presents some conclusions and future work.

2 Pattern search for bound constrained optimization

This section contains the details concerning our implementation of the Hooke and Jeeves pattern search method, in particular, the scheme used to maintain the iterates in the set Ω, the initialization of the process, and the stopping criterion. In the sequel, the following notation is used: $x_k \in \mathbb{R}^n$ denotes the approximation to the solution at the iteration k; $(x_k)_i \in \mathbb{R}$ is the i th ($i = 1, \ldots, n$) component of the point x_k; s_k is the step; Δ_k is the step length; and d_k represents a descent direction.

The Hooke and Jeeves (HJ) pattern search method has been widely used in the nonlinear programming context, emerging as an efficient algorithm for solving unconstrained, bound constrained, as well as linearly or nonlinearly nonsmooth constrained problems. It performs two types of moves: the exploratory and the pattern moves. The exploratory move carries out a coordinate search - a search along the coordinate axes -

around a selected iterate, with a step length of Δ_k. If a new iterate with a better function value is encountered, the iteration is successful. Otherwise, the iteration is unsuccessful and the step length Δ_k is reduced.

When the previous iteration was successful, the vector $x_k - x_{k-1}$ defines a promising direction and a pattern move generates a new trial iterate $x_k + (x_k - x_{k-1})$. An exploratory move is then carried out about this trial iterate rather than about the current iterate x_k. Then, if the search along the coordinates is successful, the new iterate is accepted as x_{k+1} However, if the exploratory move is unsuccessful, the pattern move is rejected and the method reduces to coordinate search around x_k [6]. To maintain feasibility in the pattern search algorithm, when x_k in not in Ω, the iterate is projected into the boundary of feasible region componentwise.

To be able to cope with variables with different scaling, our implementation of the HJ algorithm uses a vector as a step length Δ. Given an adequate initial approximation $x_1 \in \mathbb{R}^n$, each component of Δ will depend on the corresponding component of x_1, i.e., if $(x_1)_i \neq 0$ then $(\Delta_1)_i = \gamma_\Delta (x_1)_i$; otherwise $(\Delta_1)_i = \gamma_\Delta$ for $i = 1, \ldots, n$, where γ_Δ is a positive parameter.

A stopping criterion is defined to find a solution that has objective function value within a certain percentage of the optimal objective value known in the literature, f^*. For a proper termination of the algorithm when solving problems with zero optimal function values, the following conditions are used:

if $|f^*| \leq mach_\varepsilon$ **then** $|f(x_k) - f^*| \leq \varepsilon^2 |1 + f(x_k)|$ **else** $|f(x_k) - f^*| \leq \varepsilon |f(x_k)|$, where ε is a small positive constant and $mach_\varepsilon$ represents the machine zero. The algorithm also terminates if the number of objective function evaluations exceeds a maximum target $nfeval_{\max}$. The HJ pattern search algorithm can be reported through an abstract description as shown below in Algorithm 1.

Algorithm 1 Bound Constrained Pattern Search

Given $x_1 \in \Omega$; compute $f(x_1)$; set $k = 1$ and $f(x_0) = f(x_1)$
while stopping criterion is not satisfied **do**
if $f(x_{k-1}) > f(x_k)$ **then**
    pattern move
    $s_k \leftarrow$ *exploratory move*$(x_k + (x_k - x_{k-1}))$
    $x_{k+1} \leftarrow$ constrain $x_k + s_k$ in Ω
    $x_{k-1} \leftarrow x_k$
    $x_k \leftarrow x_{k+1}$
end if
if $f(x_{k-1}) \leq f(x_k)$ **then**
    $s_k \leftarrow$ *exploratory move*(x_k)
    $x_{k+1} \leftarrow$ constrain $x_k + s_k$ in Ω
end if
set $k = k + 1$
end while

In the first iteration of the process and whenever $f(x_{k-1}) \leq f(x_k)$ an exploratory move is carried out around x_k. If this move is succeeded, a pattern move follows; other-

wise an exploratory move is again carried out with a reduced step length. All the iterates generated by the algorithm should be maintained feasible.

3 Heuristic pattern search algorithms

In this section, a heuristic pattern search method is proposed for solving bound constrained optimization problems. It combines the usual pattern and exploratory moves of the HJ method with a random approximate descent search. No derivative information is required for randomly generating the descent direction, and a reduction on the number of function evaluations is expected, since some exploratory moves are replaced by a descent move. We now show how an approximate descent search can be evaluated.

Here, we describe a strategy to generate an approximate descent direction, d_k, for the objective function f, at the current iterate x_k. This is important since experience shows that search directions that are parallel to the coordinate axes may be uphill at points of the search region. Based on two points y_1 and y_2 randomly generated in the neighborhood of x_k, in such a way that $\|x_k - y_i\| \leq \varsigma$, $(i = 1,2)$ for a sufficiently small positive value of ς, a vector with a high probability of being a descent direction for the objective function at x_k is generated by

$$d_k = -\frac{1}{\sum_{j=1}^{2} |\Delta f_j|} \sum_{i=1}^{2} (\Delta f_i) \frac{x_k - y_i}{\|x_k - y_i\|}, \tag{3}$$

where $\Delta f_j = f(x_k) - f(y_j)$. Theoretical properties related to this direction vector are described in [5].

Recall that when the previous iteration was successful in the HJ moves, the pattern move defines the trial iterate $x_k + (x_k - x_{k-1})$. We now propose a heuristic pattern search algorithm that carries out an approximate descent search about that trial iterate. If $f(x_{k+1}) < f(x_k)$, for $x_{k+1} = x_k + (x_k - x_{k-1}) + \lambda d_k$ and $\lambda \in (0,1]$, the new iterate is accepted as x_{k+1}. However, if the descent move is unsuccessful, the pattern move is rejected and the method reduces to the classical coordinate search around x_k. The selection of an adequate value for the step length λ is based on the well-known backtracking line search strategy. Initially, λ is set to 1, and it is halved for at most five iterations until f is reduced. If no reduction in f is obtained, the move is considered unsuccessful. Algorithm 2 is the abstract description of the proposed framework and is denoted by heuristic pattern search (version 1).

Algorithm 3 below is an alternative implementation of the random descent search, denoted by heuristic pattern search (version 2). The procedure works as follows. After a pattern move has been carried out, the random descent move is implemented in order to find an iterate $x_{k+1} = x_k + (x_k - x_{k-1}) + \lambda d_k$ that forces a reduction in f. If f is reduced, then the new iterate is accepted; otherwise, the algorithm tries an exploratory move around $x_k + (x_k - x_{k-1})$. However, if none of these moves is successful, the pattern move is rejected and the search returns to x_k. Both random descent move and exploratory move are sequentially repeated around x_k.

Algorithm 2 Bound Constrained Heuristic Pattern Search (version 1)

Given $x_1 \in \Omega$; compute $f(x_1)$; set $k = 1$ and $f(x_0) = f(x_1)$
while stopping criterion is not satisfied **do**
if $f(x_{k-1}) > f(x_k)$ **then**
    pattern move
    $d_k \leftarrow$ *random descent move*$(x_k + (x_k - x_{k-1}))$
    $x_{k+1} \leftarrow$ constrain $x_k + (x_k - x_{k-1}) + \lambda d_k$ in Ω
    $x_{k-1} \leftarrow x_k$
    $x_k \leftarrow x_{k+1}$
end if
if $f(x_{k-1}) \leq f(x_k)$ **then**
    $s_k \leftarrow$ *exploratory move*(x_k)
    $x_{k+1} \leftarrow$ constrain $x_k + s_k$ in Ω
end if
set $k = k + 1$
end while

Algorithm 3 Bound Constrained Heuristic Pattern Search (version 2)

Given $x_1 \in \Omega$; compute $f(x_1)$; set $k = 1$ and $f(x_0) = f(x_1)$
while stopping criterion is not satisfied **do**
if $f(x_{k-1}) > f(x_k)$ **then**
    pattern move
    $d_k \leftarrow$ *random descent move*$(x_k + (x_k - x_{k-1}))$
    $x_{k+1} \leftarrow$ constrain $x_k + (x_k - x_{k-1}) + \lambda d_k$ in Ω
    if $f(x_k) \leq f(x_{k+1})$ **then**
        $s_k \leftarrow$ *exploratory move*$(x_k + (x_k - x_{k-1}))$
        $x_{k+1} \leftarrow$ constrain $x_k + s_k$ in Ω
    end if
    $x_{k-1} \leftarrow x_k$
    $x_k \leftarrow x_{k+1}$
end if
if $f(x_{k-1}) \leq f(x_k)$ **then**
    $d_k \leftarrow$ *random descent move*(x_k)
    $x_{k+1} \leftarrow$ constrain $x_k + \lambda d_k$ in Ω
    if $f(x_k) \leq f(x_{k+1})$ **then**
        $s_k \leftarrow$ *exploratory move*(x_k)
        $x_{k+1} \leftarrow$ constrain $x_k + s_k$ in Ω
    end if
    $x_{k-1} \leftarrow x_k$
    $x_k \leftarrow x_{k+1}$
end if
set $k = k + 1$
end while

4 Numerical experiments

To evaluate the performance of the herein proposed heuristic pattern search algorithms for bound constrained minimax problems, a set of 22 benchmark problems, some described in full detail in [11], and others in [13], is used. The algorithms are coded in the C programming language and contain an interface to connect to AMPL so that the problems coded in AMPL could be easily read and solved [4]. AMPL is a mathematical programming language that allows the codification of optimization problems in a powerful and easy to learn language. The set of coded problems may be obtained from the first author upon request. The list of the parameters used in the algorithms is: $mach_\varepsilon = 10^{-20}$, $\varepsilon = 10^{-4}$, $\varsigma = 10^{-3}$, $nfeval_{\max} = 20000$.

Due to the stochastic nature of the heuristic pattern search algorithms, each problem was solved 100 times. For each run, we record the solution as well as the number of iterations and the number of (objective) function evaluations. Then, f_{avg}, the average of the solutions obtained after the 100 runs, is reported. To compare the performance of the pattern search type algorithms we use the performance profiles as described in Dolan and Moré's paper [3]. The profiles are based on the metric f_{avg}. For each algorithm in comparison, the plot shows the proportion of problems in the set, denoted by $\rho(\tau)$, that has the best value of the metric, for each value of $\tau \in \mathbb{R}$. To see which algorithm gives the least value of the metric mostly, then the values of $\rho(1)$ for all the algorithms should be compared. The higher the ρ the better the solver is. On the other hand, $\rho(\tau)$ for large values of τ measures the solver robustness. We refer to [3] for details. First, we aim to analyze the effect of the parameters in the step length initialization, namely Δ and γ_Δ, on both heuristic pattern search algorithms – Algorithm 2, heuristic pattern search (version 1), and Algorithm 3, heuristic pattern search (version 2) – when compared with the original pattern search based on Hooke and Jeeves moves. When the algorithms did not find a solution with the desired accuracy, they were allowed to run for 20000 function evaluations. All problems were solved for three different values of γ_Δ: $0.01, 1, 100$. The other case in comparison sets Δ to one.

Figure 1 presents the performance profiles of the Hooke and Jeeves algorithm, for the four cases previously described. The profiles of the two proposed heuristic pattern search algorithms are shown in Figure 2 and Figure 3. We observe that the most efficient and robust initialization of the step length in the HJ algorithm is obtained when $\gamma_\Delta = 0.01$. See Figure 1. On the other hand, Algorithm 2 is more effective in reaching the most consistent results when the initial step length depends on the initial values of the variables and $\gamma_\Delta = 100$. See Figure 2. Finally, we conclude from Figure 3 that the heuristic pattern search defined by the Algorithm 3 attains the best performance mostly when $\gamma_\Delta = 100$ (observing the plot for $\tau = 1$).

We now compare the results obtained by the original pattern search method based on HJ exploratory moves with the two herein proposed heuristic pattern search algorithms. In Figure 4, a comparison based on the f value, for the deterministic pattern search, and on the f_{avg}, for the two heuristic algorithms, is presented. Clearly, the heuristic pattern search (version 2) wins over the others.

Finally, for comparative purposes, we summarize in Table 1 the average number ('average') and the standard deviation ('SD') of function evaluations ($nfeval$), obtained in [12] and [13], when solving some of the problems in our set. A unified particle

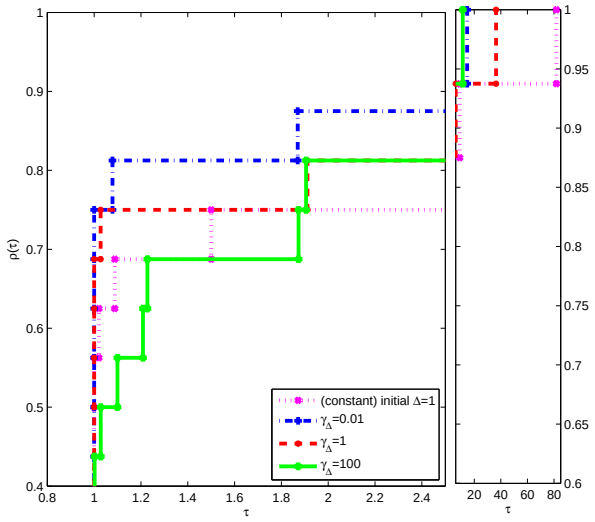

Fig. 1. Performance profile on f for the pattern search based on Hooke and Jeeves moves

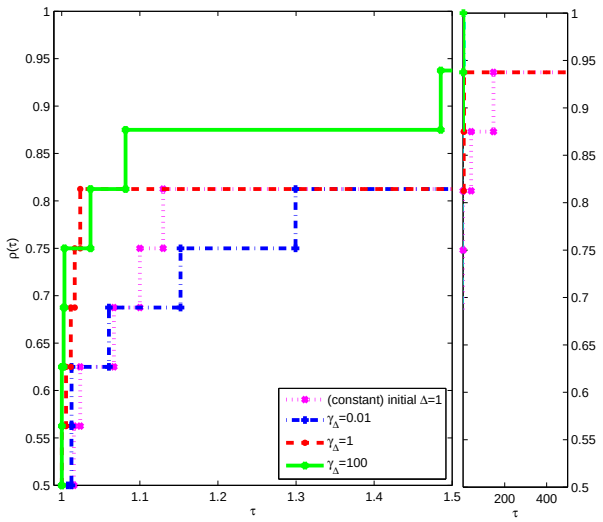

Fig. 2. Performance profile on f_{avg} for the heuristic pattern search (version 1)

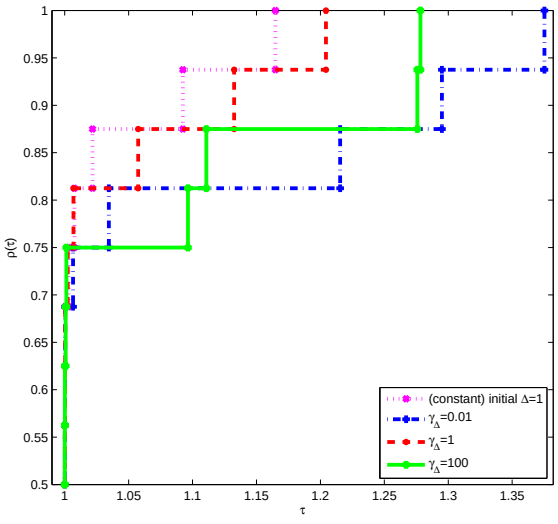

Fig. 3. Performance profile on f_{avg} for the heuristic pattern search (version 2)

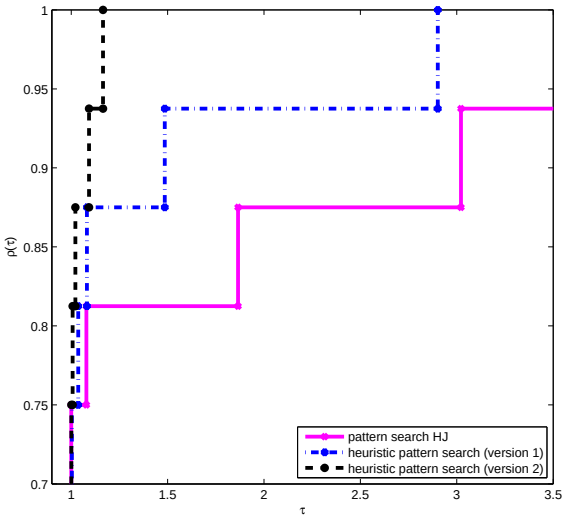

Fig. 4. Performance profile on f_{avg} for the three algorithms in comparison

swarm optimization that combines the global and local variants of the standard PSO and incorporates a stochastic parameter to imitate mutation in evolutionary algorithms is implemented in [12]. Another promising variant of the PSO, called memetic PSO, is presented in [13]. It is a hybrid algorithm that combines PSO with local search techniques. In Table 1, we report the results of its global variant. For simplicity, we report our results from the heuristic pattern search (version 2) algorithm under HPS2. The table also reports the percentage of runs that were successful, i.e., that reached the solution within an error of 10^{-4} before the 20000 function evaluations were reached. The runs that are considered unsuccessful are not used to compute the 'average' and 'SD'. Despite the problems TP17 [13], Wong 1 [11] and TP18 [13] where HPS2 reached 20000 function evaluations in some runs, the computational effort, measured by the number of function evaluations, and the % suc. (percentage of successful runs) for solving the other problems are comparable with the other methods.

Table 1. Comparison with other stochastic algorithms.

	$nfeval$ in HPS2			$nfeval$ in [12]			$nfeval$ in [13]		
Problem	average	SD	% suc.	average	SD	% suc.	average	SD	% suc.
CB2 [11]	1848.7	2619.4	99	1993.8	853.7	100	2415.3	1244.2	100
QL [11]	1809.1	2750.3	94	18294.5	2389.4	100	18520.1	776.9	100
CB3 [11]	635.8	114.3	99	1775.6	241.9	100	-	-	-
TP17 [13]	141.2	28.4	37	1670.4	530.6	100	3991.3	2545.2	100
Wong 1 [11]	283.0	123.9	64	2128.5	597.4	100	-	-	-
TP18 [13]	8948.4	5365.2	7	12801.5	5072.1	100	7021.3	1241.4	100
TP19 [13]	772.0	60.8	100	1701.6	184.9	100	2947.8	257.0	100
SPIRAL [11]	4114.7	1150.2	100	3435.5	1487.6	100	1308.8	505.5	100
OET6 [11]	324.1	173.1	100	3332.5	1775.4	100	4404.0	3308.9	100

5 Conclusions

This paper proposes and tests two algorithms that incorporate a randomly generated approximate descent search into the Hooke and Jeeves pattern search method to improve accuracy, for solving non-differentiable bound constrained optimization problems. We show that the two hybrid algorithms are able to solve bound constrained minimax problems through experiments on a set of benchmark test problems. Compared with the original Hooke and Jeeves pattern search method, the proposed hybridizations reach the solutions with higher accuracy at a reasonable computational cost. From the comparisons with other stochastic methods we observe that the proposed heuristic pattern search algorithm is competitive.

Future developments will consider the extension of these heuristic pattern search methods to solving equality and inequality constrained minimax problems, using the

test set described in [11], through the implementation of the augmented Lagrangian function described in [1] and already used in [9] in a pattern search (with equality constraints) context.

References

1. D.P. Bertsekas (1999) *Nonlinear Programming*, 2nd ed. Athena Scientific, Belmont
2. G. Di Pillo and L. Grippo (1993) A smooth method for the finite minimax problem. *Mathematical Programming*, 60, 187–214
3. E.D. Dolan and J.J. Moré (2002) Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91, 201–213
4. R. Fourer, D.M. Gay and B. Kernighan (1990) A modeling language for mathematical programing. *Management Science*, 36, 519–554 (http://www.ampl.com)
5. A.-R. Hedar and M. Fukushima (2004) Heuristic pattern search and its hybridization with simulated annealing for nonlinear global optimization. *Optimization Methods and Software*, 19, 291–308
6. R. Hooke and T.A. Jeeves (1961) Direct search solution of numerical and statistical problems. *Journal on Associated Computation*, 8, 212–229
7. E.C. Laskari, K.E. Parsopoulos and M.N. Vrahatis (2001) Particle swarm optimization for minimax problems. in *Proceedings of IEEE 2002 Congress on Evolutionary Computation*, ISBN: 0-7803-7278-6, 1576–1581
8. R.M. Lewis and V. Torczon (1999) Pattern search algorithms for bound constrained minimization. *SIAM Journal on Optimization*, 9, 1082–1099
9. R.M. Lewis and V. Torczon (2001) A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Optimization*, 12, 1075–1089
10. G. Liuzzi, S. Lucidi and M. Sciandrone (2006) A derivative-free algorithm for linearly constrained finite minimax problems. *SIAM Journal on Optimization*, 16, 1054–1075
11. L. Lukšan and J. Vlček (2000) Test problems for nonsmooth unconstrained and linearly constrained optimization. TR 798, ICS, Academy of Science of the Czech Republic, January
12. K.E. Parsopoulos and M.N. Vrahatis (2005) Unified particle swarm optimization for tackling operations research problems. in *Proceedings of IEEE 2005 Swarm Intelligence Symposium*, Pasadena, USA, 53–59
13. Y.G. Petalas, K.E. Parsopoulos and M.N. Vrahatis (2007) Memetic particle swarm optimization. *Annals of Operations Research*, 156, 99–127
14. E. Polak, J.O. Royset and R.S. Womersley (2003) Algorithms with adaptive smoothing for finite minimax problems. *Journal of Optimization Theory and Applications*, 119, 459–484
15. M.-J. Tahk, H.-W. Woo and M.-S. Park (2007) A hybrid optimization method of evolutionary and gradient search, *Engineering Optimization*, 39, 87–104
16. V. Torczon (1997) On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7, 1–25
17. S. Xu (2001) Smoothing method for minimax problems. *Computational Optimization and Applications*, 20, 267–279
18. F. Ye, H. Liu, S. Zhou and S. Liu (2008) A smoothing trust-region Newton-CG method for minimax problem. *Applied Mathmatics and Computation*, 199, 581–589
19. Z. Zhu, X. Cai and J. Jian (2009) An improved SQP algorithm for solving minimax problems. *Applied Mathmatics Letters*, 22, 464–469